УДК 004.8:004.42:378
DOI https://doi.org/10.15589/znp2026.1(503).2.6

GENERATING PEDAGOGICAL FEEDBACK FOR STUDENT CODE BASED ON LARGE LANGUAGE MODELS

ГЕНЕРАЦІЯ ПЕДАГОГІЧНОГО ЗВОРОТНОГО ЗВ'ЯЗКУ ДО СТУДЕНТСЬКОГО КОДУ НА ОСНОВІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Denys A. Seliutin
denis.selutin.ds@gmail.com
ORCID: 0009-0000-2843-9689

Д. А. Селютін,
аспірант

National Aerospace University "Kharkiv Aviation Institute", Kharkiv
Національний аерокосмічний університет «Харківський авіаційний інститут», м. Харків

Abstract. *Purpose.* The purpose of the study is to develop and substantiate a conceptual model of generating pedagogical feedback to student program code based on large language models, which provides automated analysis of program decisions, error detection and generation of meaningful recommendations to improve the effectiveness of programming training.
Method. A conceptual model of a pedagogical feedback generation system based on large language models is proposed, which integrates program code analysis, error identification, explanation generation and generation of recommendations for further improvement of the student's solution. The model involves the use of structural code analysis (AST), prompt engineering mechanisms and pedagogically oriented response templates. The architecture of an intelligent system is considered, which includes modules for collecting student decisions, code analysis, interpretation of results and generation of personalized pedagogical feedback.
Results. It is proven that traditional automatic software code evaluation systems are mostly focused on checking the correctness of the program execution result, but do not provide high-quality pedagogical explanations or recommendations for correcting errors. It is demonstrated that large language models are able to generate explanations for errors, provide recommendations for optimizing algorithms, and support the learning process by providing hints of different levels of complexity. The results obtained indicate the prospects for using large language models as a tool for intellectual support in teaching programming.
Scientific novelty. The scientific novelty lies in the development of a conceptual model of a pedagogical feedback generation system based on large language models, which combines automatic analysis of the program code, interpretation of results using LLM, and mechanisms for generating pedagogically oriented explanations and recommendations.
Practical importance. The proposed model can be used to create adaptive educational platforms that provide personalized feedback and contribute to the formation of algorithmic thinking of students. The limitations of such systems are also highlighted, in particular the possibility of generating inaccurate recommendations or erroneous explanations, which requires the use of verification mechanisms and teacher participation in the assessment process, in particular using the human-in-the-loop approach.
Key words: large language models; programming education; automatic feedback; source code analysis; artificial intelligence in education.

Анотація. *Мета.* Метою дослідження є розробка та обґрунтування концептуальної моделі генерації педагогічного зворотного зв'язку до студентського програмного коду на основі великих мовних моделей, що забезпечує автоматизований аналіз програмних рішень, виявлення помилок і формування змістовних рекомендацій для підвищення ефективності навчання програмуванню.
Методика. Запропоновано концептуальну модель системи генерації педагогічного зворотного зв'язку на основі великих мовних моделей, яка інтегрує аналіз програмного коду, ідентифікацію помилок, формування пояснень та генерацію рекомендацій для подальшого вдосконалення рішення студентом. Модель передбачає використання структурного аналізу коду (AST), механізмів промпт-інжинірингу та педагогічно орієнтованих шаблонів відповіді. Розглянуто архітектуру інтелектуальної системи, що включає модулі збору студентських рішень, аналізу коду, інтерпретації результатів та генерації персоналізованого педагогічного зворотного зв'язку.

Результати. Доведено, що традиційні системи автоматичного оцінювання програмного коду здебільшого орієнтовані на перевірку правильності результату виконання програми, але не забезпечують якісних педагогічних пояснень або рекомендацій щодо виправлення помилок. Продемонстровано, що великі мовні моделі здатні формувати пояснення до помилок, надавати рекомендації щодо оптимізації алгоритмів і підтримувати процес навчання шляхом надання підказок різних рівнів складності. Отримані результати свідчать про перспективність використання великих мовних моделей як інструменту інтелектуальної підтримки викладання програмування.
Наукова новизна. Наукова новизна полягає у розробці концептуальної моделі системи генерації педагогічного зворотного зв'язку на основі великих мовних моделей, яка поєднує автоматичний аналіз програмного коду, інтерпретацію результатів за допомогою LLM та механізми формування педагогічно орієнтованих пояснень і рекомендацій.
Практична значущість. Запропонована модель може бути використана для створення адаптивних освітніх платформ, які забезпечують персоналізований зворотний зв'язок і сприяють формуванню алгоритмічного мислення студентів. Також виділено і обмеження таких систем, зокрема можливість генерації неточних рекомендацій або помилкових пояснень, що потребує використання механізмів перевірки та участі викладача у процесі оцінювання, зокрема із застосуванням підходу human-in-the-loop.
Ключові слова: великі мовні моделі; програмна освіта; автоматичний зворотний зв'язок; аналіз програмного коду; штучний інтелект у освіті.

ПОСТАНОВКА ЗАДАЧІ

Навчання програмуванню є одним із найскладніших етапів підготовки майбутніх фахівців у галузі інформаційних технологій. Однією з ключових умов ефективного навчання є своєчасний та змістовний педагогічний зворотний зв'язок щодо студентських програмних рішень. Саме зворотний зв'язок дозволяє студентам усвідомити власні помилки, зрозуміти логіку алгоритмів та покращити якість програмного коду.

У сучасних умовах масової освіти викладачі стикаються з проблемою значного навантаження під час перевірки програмних завдань. Це особливо актуально для курсів програмування, де кожне рішення потребує аналізу логіки алгоритму, структури коду та правильності реалізації. У зв'язку з цим активно розвиваються системи автоматичного оцінювання програмного коду (autograding systems).

Традиційні системи автоматичного оцінювання здебільшого перевіряють правильність виконання програми за допомогою тестових наборів даних або статичного аналізу коду. Проте такі системи не забезпечують повноцінного педагогічного зворотного зв'язку, оскільки часто обмежуються повідомленням про помилки або результат тестування.

З появою великих мовних моделей (LLM), таких як GPT-подібні системи, з'явилася можливість автоматично генерувати пояснення, підказки та рекомендації щодо виправлення помилок у програмному коді. Такі моделі здатні аналізувати текст програм, інтерпретувати логіку алгоритмів та формувати природномовні пояснення для студентів.

Дослідження показують, що LLM можуть генерувати індивідуалізований зворотний зв'язок для студентських програмних рішень, допомагаючи студентам краще розуміти власні помилки та покращувати програмні навички.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Проблема формування якісного педагогічного зворотного зв'язку до студентського програмного коду тривалий час розглядалася переважно в межах автоматизованого оцінювання програмних вправ. У фундаментальному огляді Г. Кеунінга, Дж. Йорінга, Б. Герена показано, що більшість ранніх систем автоматичного фідбеку орієнтувалися на виявлення синтаксичних і функціональних помилок, тоді як пояснювальний, адаптивний і дидактично вмотивований зворотний зв'язок реалізовувався значно рідше [1]. Подальший розвиток цієї проблематики Дж. Йьорінг, Х. Кеунінг, С. Марван та ін. пов'язали з потребою у своєчасному формувальному фідбеку та підказках для початківців-програмістів, наголосивши, що ефективний супровід має враховувати не лише кінцевий результат, а й проміжні кроки розв'язання задачі [2].

Сучасний стан автоматизованого оцінювання програмування узагальнено у системному огляді М. Мессера, Н. К. Брауна, М. Кьоллінга, М. Ши, які проаналізували 121 працю та підтвердили, що провідними залишаються підходи, засновані на модульному тестуванні, порівнянні з еталонними розв'язками та статичному аналізі коду [3]. Автори підкреслюють, що такі інструменти добре масштабуються, проте їхній зворотний зв'язок часто є обмеженим: він повідомляє, що саме не пройшло перевірку, але рідко пояснює, чому виникла помилка і як студенту покращити власне алгоритмічне мислення. Саме ця обставина створила передумови для інтеграції великих мовних моделей у програмну освіту [3].

Одними з перших робіт, що продемонстрували практичний потенціал LLM у підтримці студентів-програмістів, стали дослідження М. Ліфітона, Б. Шиза, Дж. Савелка, П. Денні, Н. Кіслера, Д. Лора,

Х. Кеунінга тощо. Так, у праці про систему CodeHelp показано, що великі мовні моделі можуть бути використані як інструмент масштабованої підтримки в курсах програмування за умови запровадження «guardrails», тобто обмежень, які не дозволяють моделі просто видавати готове рішення [4]. У той же час Н. Кіслер, Д. Лор та Х. Кеунінг експериментально підтвердили, що LLM здатні генерувати формувальний зворотний зв'язок до вступних задач з програмування, однак його якість суттєво залежить від типу помилки, формулювання запиту та рівня підготовки користувача [5]. Отже, вже на початковому етапі досліджень було встановлено, що LLM доцільно розглядати не як повну заміну викладача, а як інтелектуальний інструмент підтримки навчання.

Важливим етапом стали дослідження, у яких LLM використовувалися безпосередньо для генерації фідбеку до реальних студентських розв'язків. С. Джейкобс та С. Яшке описали застосування GPT – 4 у вебсистемі для програмної освіти й показали, що модель у більшості випадків коректно реагує на помилки у коді, не розкриваючи одразу повного розв'язку, хоча проблема «галюцинацій» та неточних підказок залишається суттєвою [6]. Схожі висновки отримали І. Азаїз, Н. Кіслер та С. Стрікрот, які на автентичних студентських поданнях продемонстрували, що GPT – 4 Turbo здатна формувати більш структурований, персоналізований і локалізований фідбек, ніж попередні моделі, однак інколи генерує суперечливі повідомлення, наприклад одночасно визнає розв'язок правильним і пропонує його виправляти [7]. Таким чином, окреслені дослідження дозволили довести високу евристичну цінність LLM-фідбеку, але також виявляють потребу у додаткових механізмах валідації відповідей.

Подальший розвиток напряму пов'язаний із переходом від окремих експериментів до системного оцінювання ефективності LLM у різних освітніх контекстах. І. Естевес-Айрес, П. Каллехо, М. Хомбрадос-Еррера, К. Аларіо-Ойос, К. Дельгадо Клос дослідили використання чат-ботів для генерування фідбеку в курсі з паралельного програмування та показали, що моделі можуть бути корисними навіть у тематиках із підвищеною концептуальною складністю, зокрема для виявлення deadlock, starvation і race condition [8]. У свою чергу М. Юсеф, К. Мохамед, В. Медхат, Е. Мохамед, Г. Хоріба та Т. Арафа у системі BeGrading розглянули LLM вже не тільки як інструмент підказок, але і як компонент розширеного циклу оцінювання й фідбеку в програмній освіті, акцентуючи увагу на підвищенні швидкості та послідовності оцінювання [9]. Окремо слід відзначити роботу Е. Чжу, С. Теджа, К. Кумбса та Д. Паттерсона, у якій система FEEDBOT зорієнтована на формувальний дизайн-фідбек до програмних завдань, тобто на підтримку не лише правильності коду, а й якості проєктних рішень [10]. Це свідчить про поступове розширення функцій LLM від локального коментування помилок до комплексної педагогічної підтримки.

Аналіз новітніх праць дозволяє констатувати перехід до адаптивних та аналітичних моделей використання LLM у програмній освіті. Д. Кім, С. Кім, Й. Джо у моделі SQKT інтегрують кодові подання, студентські запитання та GPT-базоване виділення навичок для прогнозування подальшої успішності студента, тобто пов'язують мовні моделі не тільки з фідбеком, а й з knowledge tracing та персоналізацією навчання [11]. У спеціальному огляді LLM-застосувань у програмній освіті Г. Піттс, А. Гріді, А. Лекшмі Нараянан систематизували дослідження за трьома напрямами (формувальний фідбек до коду, оцінювання та моделювання знань) та дійшли до висновку, що найбільш перспективними є рішення з human-in-the-loop, де експертний контроль доповнює автоматичну генерацію [12].

Отже, результати аналізу сучасних публікацій дозволяють стверджувати, що науковий фокус зміщується від простого автоматичного оцінювання коду до побудови педагогічно виважених, адаптивних та пояснювальних систем на основі великих мовних моделей.

ВІДОКРЕМЛЕННЯ НЕВИРІШЕНИХ РАНІШЕ ЧАСТИН ЗАГАЛЬНОЇ ПРОБЛЕМИ

На основі здійсненого аналізу джерельної бази, присвяченої використанню великих мовних моделей у програмній освіті, можна зробити висновок, що ряд аспектів цієї проблематики залишається недостатньо опрацьованою. Так, наприклад, більшість існуючих систем автоматичного оцінювання програмного коду зосереджуються на перевірці правильності виконання програм або виявленні синтаксичних і функціональних помилок, тоді як формування змістовного педагогічного зворотного зв'язку, спрямованого на пояснення логіки помилки та підтримку розвитку алгоритмічного мислення студента, реалізується досить обмежено. Недостатньо дослідженими також залишаються питання системної організації процесу генерації педагогічного фідбеку, зокрема поєднання автоматичного аналізу програмного коду, інтерпретації результатів за допомогою LLM та формування педагогічно орієнтованих пояснень, адаптованих до рівня підготовки студента. У багатьох наведених дослідження підкреслюється проблема можливих неточностей бо суперечностей у відповідях мовних моделей, що зумовлює потребу у впровадженні механізмів перевірки та педагогічного контролю.

Тобто можна відзначити, що невирішеною частиною загальної проблеми є розробка цілісної концептуальної моделі системи генерації педагогічного зворотного зв'язку до студентського програмного коду на основі великих мовних моделей, яка б забезпечувала інтеграцію аналізу програмних рішень, виявлення

помилок, генерації пояснень і рекомендацій, а також враховувала педагогічні вимоги до формувального фідбеку в процесі навчання програмуванню.

МЕТА ДОСЛІДЖЕННЯ

Метою дослідження є розробка та обґрунтування концептуальної моделі генерації педагогічного зворотного зв'язку до студентського програмного коду на основі великих мовних моделей, що забезпечує автоматизований аналіз програмних рішень, виявлення помилок і формування змістовних рекомендацій для підвищення ефективності навчання програмуванню.

МЕТОДИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об'єкт дослідження – процес формування педагогічного зворотного зв'язку під час навчання програмуванню у закладах вищої освіти. **Предмет дослідження** – моделі, методи та архітектурні рішення генерації педагогічного зворотного зв'язку до студентського програмного коду на основі великих мовних моделей, що забезпечують автоматизований аналіз програмних рішень, виявлення помилок та формування змістовних рекомендацій.

МЕТОДИ ДОСЛІДЖЕННЯ

Для досягнення поставленої мети використано наступні методів:

– аналіз і узагальнення наукових джерел для виявлення сучасних підходів до автоматизованого оцінювання програмного коду та застосування великих мовних моделей у програмній освіті;

– системний аналіз з метою визначення структурних компонентів інтелектуальної системи генерації педагогічного зворотного зв'язку та встановлення взаємозв'язків між ними;

– метод концептуального моделювання – був використаний для розробки моделі системи генерації педагогічного зворотного зв'язку на основі LLM;

– порівняльний аналіз – зіставлення традиційних систем автоматичного оцінювання програмного коду та підходів із використанням великих мовних моделей;

– структурний аналіз програмного коду (AST-підхід) з метою формалізації процесу виявлення помилок і підготовки даних для подальшої інтерпретації мовною моделлю.

ОСНОВНИЙ МАТЕРІАЛ

Концептуальна модель генерації педагогічного зворотного зв'язку до студентського програмного коду ґрунтується на інтеграції методів автоматичного аналізу програм, інтелектуальної обробки даних та можливостей великих мовних моделей для формування змістовних пояснень і рекомендацій у процесі навчання програмуванню. Запропонована система передбачає послідовну обробку студентських програмних рішень та формування педагогічно орієнтованого зворотного зв'язку, який спрямований не лише на фіксацію помилок, але й на підтримку навчального процесу шляхом пояснення причин їх виникнення та надання рекомендацій щодо вдосконалення алгоритмічних рішень. Узагальнену архітектуру запропонованої системи подано на рисунку 1.

Функціональна структура системи включає кілька взаємопов'язаних модулів, кожен з яких виконує окремі етапи обробки студентського програмного коду. Початковим елементом архітектури є модуль збору даних, який забезпечує прийом студентських програмних рішень, їх збереження та подальшу передачу до модулів аналізу. У межах цього модуля здійснюється інтеграція з електронними освітніми платформами та системами дистанційного навчання, такими як LMS або системи автоматичного оцінювання програмних завдань. Окрім цього, модуль накопичує історію виконаних студентом завдань, що дозволяє формувати базу даних навчальних рішень та використовувати її для подальшого аналізу прогресу студента.

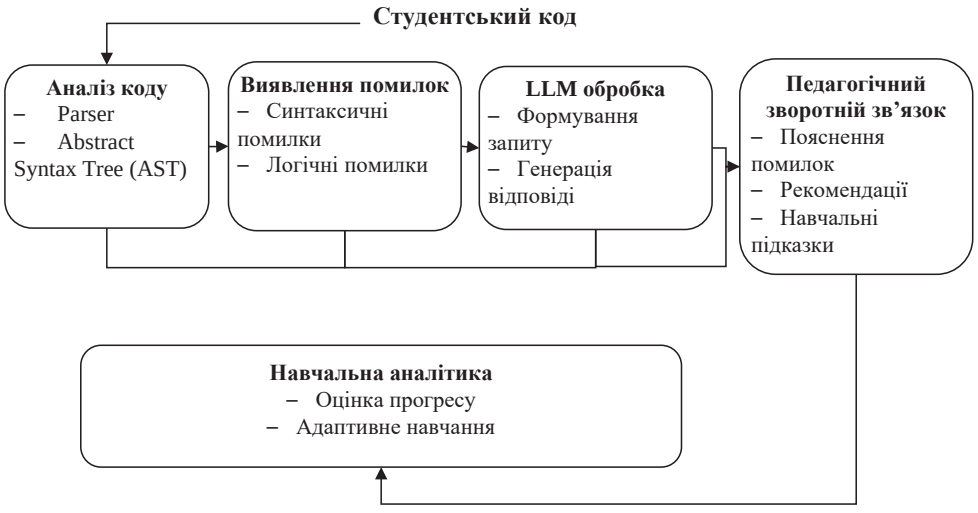

Рис. 1. Архітектура pipeline генерації педагогічного зворотного зв'язку до студентського коду на основі великих мовних моделей

Наступним етапом функціонування системи є модуль аналізу програмного коду, який виконує синтаксичну та структурну обробку отриманих програм. На цьому етапі здійснюється парсинг програмного коду, побудова абстрактного синтаксичного дерева (Abstract Syntax Tree, AST), а також виявлення типових синтаксичних помилок і порушень структури програми. Формування AST дозволяє перейти від поверхневого текстового аналізу до структурного представлення програмного коду, що значно підвищує точність подальшого аналізу та дозволяє ідентифікувати типові шаблони помилок, характерні для студентських програмних рішень.

Після первинного аналізу коду інформація передається до аналітичного модуля, який здійснює більш глибоку інтерпретацію програмного рішення. Основним завданням цього модуля є виявлення логічних помилок, які не можуть бути визначені лише на рівні синтаксичного аналізу. До таких помилок належать некоректна реалізація алгоритмічних конструкцій, неправильне використання циклів або умовних операторів, а також помилки у роботі з даними. Окрім цього, у межах аналітичного модуля виконується оцінювання алгоритмічної складності запропонованого рішення та його порівняння з еталонними реалізаціями, що дозволяє виявити можливі оптимізації або альтернативні підходи до розв'язання задачі. Результати такого аналізу формують структурований опис помилок і характеристик програмного рішення, який надалі використовується для генерації педагогічного зворотного зв'язку.

Центральним компонентом запропонованої системи є модуль генерації педагогічного зворотного зв'язку, який використовує можливості великих мовних моделей для формування зрозумілих і педагогічно орієнтованих пояснень. На основі результатів попереднього аналізу цей модуль генерує текстові коментарі до програмного коду, пояснює причини виникнення помилок та пропонує рекомендації щодо їх виправлення. Важливою функцією модуля є також формування навчальних підказок різних рівнів складності, що дозволяє підтримувати поступове засвоєння навчального матеріалу. Зокрема, система може надавати як загальні рекомендації щодо напрямку виправлення помилки, так і більш детальні пояснення алгоритмічної логіки або приклади альтернативних підходів до розв'язання задачі.

Завершальним елементом архітектури є модуль адаптації навчання, який забезпечує персоналізацію згенерованого зворотного зв'язку з урахуванням індивідуальних особливостей студента. У межах цього модуля здійснюється аналіз історії виконання програмних завдань, оцінювання динаміки навчального прогресу та визначення типових труднощів, з якими стикається студент у процесі навчання. На основі цих даних система може формувати індивідуальні рекомендації, адаптувати рівень складності підказок та пропонувати додаткові навчальні матеріали. Таким чином, модуль адаптації забезпечує перехід від статичного зворотного зв'язку до адаптивної освітньої підтримки, орієнтованої на індивідуальну траєкторію навчання.

Узагальнюючи, запропонована концептуальна модель інтегрує інструменти автоматичного аналізу програмного коду, методи інтелектуальної обробки даних та генеративні можливості великих мовних моделей для формування педагогічно змістовного зворотного зв'язку. Такий підхід дозволяє автоматизувати процес оцінювання програмних завдань, підвищити якість навчального супроводу та забезпечити більш ефективну підтримку студентів у процесі опанування програмування.

Архітектура інтелектуальної системи генерації педагогічного зворотного зв'язку до студентського програмного коду базується на принципах модульності, послідовної обробки даних та інтеграції інструментів автоматичного аналізу програм із можливостями великих мовних моделей. Функціонування системи може бути представлене у вигляді узагальненого обчислювального конвеєра (pipeline), у межах якого студентський програмний код проходить кілька послідовних етапів обробки: від первинного аналізу структури програми до генерації педагогічно орієнтованого зворотного зв'язку та подальшої підтримки навчального процесу. У загальному вигляді цей процес можна описати як послідовність: Student Code → Code Analysis → LLM Processing → Pedagogical Feedback → Learning Support, що відображає логіку перетворення студентського програмного рішення на інформативні дидактичні рекомендації.

Початковим етапом функціонування архітектури є обробка студентського програмного коду, який надходить до системи через інтерфейс навчальної платформи або автоматизованої системи оцінювання програмних завдань. На цьому етапі код передається до компонента Code Parser, який здійснює синтаксичний аналіз програмного тексту та формує його структурне представлення. Основним завданням цього компонента є розпізнавання синтаксичних конструкцій мови програмування, побудова абстрактного синтаксичного дерева та виділення ключових структурних елементів програми, таких як функції, змінні, цикли, умовні оператори та виклики процедур. Використання структурного представлення програмного коду дозволяє перейти від поверхневого текстового аналізу до більш глибокого аналізу логіки програми, що є необхідною умовою для подальшого виявлення помилок і формування змістовного педагогічного фідбеку.

Результати синтаксичного аналізу передаються до компонента Error Detector, який виконує ідентифікацію помилок у програмному коді. Цей компонент

реалізує комплекс процедур статичного та частково динамічного аналізу програм і спрямований на виявлення як синтаксичних, так і логічних помилок. До синтаксичних помилок належать порушення правил мови програмування, некоректне використання операторів або неправильна структура програми. Логічні помилки, у свою чергу, пов'язані з некоректною реалізацією алгоритмічної логіки, неправильним використанням умовних конструкцій, помилками в організації циклів або обробці даних. У процесі аналізу також можуть використовуватися еталонні рішення або набір тестових прикладів, що дозволяє визначити відповідність програмного рішення поставленому завданню.

Після етапу виявлення помилок сформована інформація передається до компонента LLM Engine, який є ключовим елементом інтелектуальної архітектури системи. У межах цього модуля результати аналізу програмного коду перетворюються на структурований контекст, що використовується великими мовними моделями для генерації пояснень. На відміну від традиційних систем автоматичного оцінювання, які здебільшого повідомляють лише факт наявності помилки, використання LLM дозволяє сформувати розгорнуте текстове пояснення, яке описує причини виникнення помилки, пояснює її вплив на роботу програми та пропонує можливі способи її виправлення. Завдяки використанню мовних моделей система здатна інтерпретувати результати технічного аналізу коду та перетворювати їх у зрозумілий для студента педагогічний коментар.

Згенеровані пояснення передаються до компонента Feedback Generator, який відповідає за формування структурованого педагогічного зворотного зв'язку. У межах цього компонента відбувається інтеграція результатів аналізу коду, пояснень, отриманих від мовної моделі, та дидактичних рекомендацій, спрямованих на покращення програмного рішення. Система може формувати різні типи фідбеку, зокрема коментарі до окремих фрагментів коду, рекомендації щодо оптимізації алгоритму, підказки щодо можливих альтернативних підходів до розв'язання задачі, а також загальні рекомендації щодо стилю програмування. Особливістю цього етапу є орієнтація не лише на виправлення конкретної помилки, але й на розвиток алгоритмічного мислення та програмної культури студента.

Завершальним елементом архітектури є Learning Analytics Module, який забезпечує аналіз навчальної діяльності студента та адаптацію педагогічного зворотного зв'язку до його індивідуальних особливостей. У цьому модулі накопичується інформація про історію виконання студентом програмних завдань, частоту виникнення певних типів помилок та динаміку навчального прогресу. Аналіз таких даних дозволяє визначати типові труднощі у процесі навчання програмуванню та формувати персоналізовані рекомендації для подальшого вдосконалення знань і навичок. Таким чином, система переходить від одноразового фідбеку до довготривалої підтримки навчального процесу, забезпечуючи адаптивний супровід студента у процесі опанування програмування.

У цілому запропонована архітектура поєднує можливості автоматичного аналізу програмного коду, інтелектуальної генерації текстових пояснень та освітньої аналітики, що дозволяє створити ефективний інструмент підтримки програмної освіти. Використання конвеєрної моделі обробки даних забезпечує масштабованість системи, її інтеграцію з сучасними освітніми платформами та можливість подальшого розширення функціональних можливостей, зокрема шляхом використання адаптивних моделей навчання та більш складних алгоритмів аналізу програмних рішень.

ОБГОВОРЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Інтеграція великих мовних моделей у процес навчання програмуванню відкриває нові можливості для підвищення ефективності педагогічного супроводу студентських програмних завдань. На відміну від традиційних систем автоматичного оцінювання програмного коду, які здебільшого орієнтовані на перевірку правильності виконання програм через тестові набори або формальні правила аналізу коду, великі мовні моделі дозволяють реалізувати більш гнучкий та педагогічно орієнтований підхід до формування зворотного зв'язку. Використання таких моделей забезпечує автоматизацію значної частини процесу перевірки програмних завдань, що є особливо важливим у сучасних умовах масової освіти, де кількість студентських робіт значно перевищує можливості індивідуального аналізу з боку викладача.

Однією з ключових переваг застосування великих мовних моделей є можливість формування персоналізованого педагогічного зворотного зв'язку. На основі аналізу конкретного програмного рішення система може генерувати індивідуальні пояснення до виявлених помилок, враховуючи структуру коду, використані алгоритмічні конструкції та контекст навчального завдання. Такий підхід дозволяє адаптувати зміст рекомендацій до рівня підготовки студента, що сприяє більш ефективному засвоєнню навчального матеріалу та формуванню навичок програмування. Крім того, мовні моделі здатні генерувати пояснення природною мовою, що робить зворотний зв'язок більш зрозумілим і доступним для студентів, особливо на початкових етапах навчання, коли формальні повідомлення про помилки часто є складними для інтерпретації.

Важливою перевагою використання LLM є також можливість генерації навчальних підказок різного рівня деталізації. На відміну від традиційних систем автоматичного оцінювання, які повідомляють лише про наявність помилки, інтелектуальна система на

основі мовних моделей може поступово спрямовувати студента до правильного рішення, пропонуючи рекомендації щодо аналізу алгоритму, пояснюючи логіку виконання програми або вказуючи на потенційні шляхи оптимізації коду. Такий підхід відповідає принципам формувального навчання, відповідно до яких зворотний зв'язок повинен не лише фіксувати результат, але й підтримувати процес розвитку когнітивних та аналітичних навичок студента.

Застосування великих мовних моделей також створює умови для масштабування освітніх курсів, що є важливим фактором у сучасній цифровій освіті. Завдяки автоматичній генерації пояснень і рекомендацій система може забезпечувати педагогічний супровід великої кількості студентів без значного збільшення навантаження на викладача. Це особливо актуально для масових онлайн-курсів та програм дистанційного навчання, де індивідуальна взаємодія між викладачем і студентом часто є обмеженою.

Результати сучасних досліджень свідчать, що згенерований великими мовними моделями зворотний зв'язок у багатьох випадках може мати якість, порівнянну з коментарями викладачів, особливо щодо ясності формулювання, логічної структури пояснень та здатності інтерпретувати помилки у програмному коді. Завдяки своїй здатності аналізувати текстові структури та контекст програмного коду мовні моделі можуть формувати коментарі, які не лише описують помилку, але й пояснюють її вплив на виконання програми та пропонують можливі шляхи її виправлення. Таким чином, використання великих мовних моделей створює передумови для формування більш інтелектуальних систем підтримки програмної освіти, здатних поєднувати автоматичний аналіз програмного коду з педагогічно обґрунтованим зворотним зв'язком.

Визнаючи значний потенціал застосування великих мовних моделей у процесі навчання програмуванню та генерації педагогічного зворотного зв'язку, необхідно відзначити, що використання таких технологій пов'язане з низкою методологічних, технічних і педагогічних обмежень. Однією з ключових проблем є можливість генерації неточних або частково некоректних рекомендацій. Великі мовні моделі, навіть за наявності значних обсягів навчальних даних, не завжди здатні гарантувати повну коректність сформованих пояснень, оскільки їхня робота базується на ймовірнісних механізмах генерації тексту. У результаті система може формувати пояснення, які виглядають логічними та переконливими, але містять неточності або не повністю відповідають конкретній логіці виконання програмного коду. У контексті освітнього процесу це може призводити до формування у студентів хибних уявлень про алгоритмічні принципи або механізми роботи програм.

Ще одним важливим викликом є складність перевірки коректності згенерованих пояснень. На відміну від традиційних систем автоматичного оцінювання, де результати базуються на формальних правилах тестування або статичного аналізу коду, пояснення, сформовані мовними моделями, мають текстову форму і часто містять інтерпретаційні елементи. Це ускладнює автоматичну валідацію таких повідомлень та потребує додаткових механізмів контролю якості. Зокрема, у складних програмних задачах модель може пропонувати рекомендації, які формально виглядають коректними, але не враховують специфіку поставленого завдання або обмеження конкретної мови програмування.

Окремою проблемою є ризик надмірної автоматизації навчального процесу. Надмірне покладання на автоматично згенерований фідбек може знизити роль активної взаємодії між студентом і викладачем, що є важливим елементом формування глибокого розуміння алгоритмічних принципів і програмних конструкцій. У випадку некритичного використання інтелектуальних систем студенти можуть сприймати автоматичні рекомендації як остаточне рішення, не аналізуючи їхню логіку та не розвиваючи власні навички аналізу програмного коду. Таким чином, використання LLM у навчальному процесі потребує ретельного педагогічного балансування між автоматизованою підтримкою та традиційними методами навчання.

Важливим фактором, що впливає на ефективність роботи систем на основі великих мовних моделей, є також залежність якості згенерованого зворотного зв'язку від формулювання запитів до моделі та якості навчальних даних. Процес формування запитів, відомий як промпт-інжиніринг, відіграє ключову роль у забезпеченні релевантності та точності отриманих пояснень. Невдало сформульовані запити можуть призводити до генерації занадто загальних або поверхневих рекомендацій, які не враховують конкретні особливості програмного коду. Крім того, результати роботи мовних моделей значною мірою залежать від характеру даних, на яких вони були навчальні, що може впливати на точність інтерпретації окремих програмних конструкцій або алгоритмічних підходів.

З огляду на зазначені обмеження, у сучасних дослідженнях дедалі частіше застосовується концепція human-in-the-loop, яка передбачає поєднання можливостей штучного інтелекту з експертним контролем з боку викладача або навчальної системи [12]. У межах цієї концепції великі мовні моделі використовуються як інструмент попереднього аналізу та генерації пояснень, тоді як остаточна оцінка та коригування рекомендацій можуть здійснюватися людиною-експертом. Така модель дозволяє поєднати переваги автоматизації з педагогічною експертизою викладача, забезпечуючи баланс між ефективністю технологічних рішень та якістю освітнього процесу.

У результаті використання підходу human-in-the-loop сприяє підвищенню надійності інтелектуальних систем підтримки навчання та зменшує ризики, пов'язані з використанням автоматично згенерованого педагогічного зворотного зв'язку.

ВИСНОВКИ

У статті досліджено можливості використання великих мовних моделей для генерації педагогічного зворотного зв'язку до студентського програмного коду. Проведений аналіз сучасних наукових публікацій показав, що інтеграція технологій штучного інтелекту у програмну освіту створює нові можливості для підвищення ефективності навчального процесу, зокрема шляхом автоматизації перевірки програмних завдань, формування змістовних пояснень до помилок та надання персоналізованих рекомендацій студентам. На основі узагальнення існуючих підходів запропоновано концептуальну модель системи генерації педагогічного зворотного зв'язку, яка поєднує автоматичний аналіз програмного коду, інтерпретацію результатів за допомогою великих мовних моделей та механізми навчальної аналітики. Розроблена архітектура системи реалізує послідовний конвеєр обробки студентського програмного рішення, що включає етапи структурного аналізу коду, виявлення помилок, генерації пояснень та формування педагогічно орієнтованого зворотного зв'язку.

Показано, що використання великих мовних моделей дозволяє перейти від традиційних систем автоматичного оцінювання програмного коду до інтелектуальних систем підтримки навчання, здатних не лише фіксувати результат виконання завдання, але й сприяти розвитку алгоритмічного мислення студентів шляхом пояснення логіки помилок та пропозиції альтернативних підходів до розв'язання задачі. Водночас встановлено, що застосування LLM у програмній освіті супроводжується низкою викликів, пов'язаних із можливістю генерації неточних рекомендацій, складністю перевірки коректності пояснень та необхідністю збереження балансу між автоматизацією навчального процесу і педагогічною взаємодією між студентом і викладачем. У зв'язку з цим доцільним є використання підходу human-in-the-loop, який передбачає поєднання можливостей штучного інтелекту з експертним контролем з боку викладача.

Перспективи подальших досліджень можуть бути пов'язані з розробкою методів оцінювання якості згенерованого педагогічного зворотного зв'язку, удосконаленням алгоритмів аналізу програмного коду та інтеграцією великих мовних моделей із системами навчальної аналітики. Важливим напрямом є створення адаптивних освітніх систем, здатних враховувати індивідуальні особливості навчання студентів та формувати персоналізовані рекомендації на основі аналізу їхнього програмного прогресу. Перспективним також уявляється дослідження гібридних моделей інтелектуальної підтримки програмної освіти, які поєднують автоматичні методи аналізу програмного коду, можливості генеративних моделей штучного інтелекту та експертні знання викладачів.

REFERENCES

[1] Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), Article 3, 1–43. DOI: https://doi.org/10.1145/3231711

[2] Jeuring, J., Keuning, H., Marwan, S., Bouvier, D., Izu, C., Kiesler, N., Lehtinen, T., Lohr, D., Peterson, A., & Sarsa, S. (2022). Towards giving timely formative feedback and hints to novice programmers. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 95–115). DOI: https://doi.org/10.1145/3571785.3574124

[3] Messer, M., Brown, N. C. C., Kölling, M., & Shi, M. (2024). Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Computing Education, 24*(1), 1–43. DOI: https://doi.org/10.1145/3636515

[4] Liffiton, M. H., Sheese, B. E., Savelka, J., & Denny, P. (2023). CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Article 8, pp. 1–11). DOI: https://doi.org/10.1145/3631802.3631830

[5] Kiesler, N., Lohr, D., & Keuning, H. (2024). Exploring the potential of large language models to generate formative programming feedback. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). DOI: https://doi.org/10.1109/FIE58773.2023.10343457

[6] Jacobs, S., & Jaschke, S. (2024). Evaluating the application of large language models to generate feedback in programming education. In *2024 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1–5). DOI: https://doi.org/10.1109/EDUCON60312.2024.10578838

[7] Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-generation for programming exercises with GPT-4. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* (Vol. 1, pp. 31–37). DOI: https://doi.org/10.1145/3649217.3653594

[8] Estévez-Ayres, I., Callejo, P., Hombrados-Herrera, M. Á., Alario-Hoyos, C., & Delgado Kloos, C. (2025). Evaluation of LLM tools for feedback generation in a course on concurrent programming. *International Journal of Artificial Intelligence in Education, 35*, 774–790. DOI: https://doi.org/10.1007/s40593-024-00406-0

[9] Yousef, M., Mohamed, K., Medhat, W., Mohamed, E. H., Khoriba, G., & Arafa, T. (2025). BeGrading: Large language models for enhanced feedback in programming education. *Neural Computing and Applications, 37*(2), 1027–1040. DOI: https://doi.org/10.1007/s00521-024-10449-y

[10] Zhu, E., Teja, S., Coombes, C., & Patterson, D. (2025). FEEDBOT: Formative design feedback on programming assignments. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education* (pp. 576–582). DOI: https://doi.org/10.1145/3724363.3729063

[11] Kim, D., Kim, S., & Jo, Y. (2025). Knowledge tracing in programming education integrating students' questions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 27703–27718). DOI: https://doi.org/10.18653/v1/2025.acl-long.1343

[12] Pitts, G., Hridi, A. P., & Lekshmi Narayanan, A. B. (2025). A survey of LLM-based applications in programming education: Balancing automation and human oversight. In *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+NLP)* (pp. 255–262). DOI: https://doi.org/10.18653/v1/2025.hcinlp-1.21

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), Article 3, 1–43. DOI: https://doi.org/10.1145/3231711

[2] Jeuring, J., Keuning, H., Marwan, S., Bouvier, D., Izu, C., Kiesler, N., Lehtinen, T., Lohr, D., Peterson, A., & Sarsa, S. (2022). Towards giving timely formative feedback and hints to novice programmers. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 95–115). DOI: https://doi.org/10.1145/3571785.3574124

[3] Messer, M., Brown, N. C. C., Kölling, M., & Shi, M. (2024). Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Computing Education, 24*(1), 1–43. DOI: https://doi.org/10.1145/3636515

[4] Liffiton, M. H., Sheese, B. E., Savelka, J., & Denny, P. (2023). CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (Article 8, pp. 1–11). DOI: https://doi.org/10.1145/3631802.3631830

[5] Kiesler, N., Lohr, D., & Keuning, H. (2024). Exploring the potential of large language models to generate formative programming feedback. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). DOI: https://doi.org/10.1109/FIE58773.2023.10343457

[6] Jacobs, S., & Jaschke, S. (2024). Evaluating the application of large language models to generate feedback in programming education. In *2024 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1–5). DOI: https://doi.org/10.1109/EDUCON60312.2024.10578838

[7] Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-generation for programming exercises with GPT-4. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* (Vol. 1, pp. 31–37). DOI: https://doi.org/10.1145/3649217.3653594

[8] Estévez-Ayres, I., Callejo, P., Hombrados-Herrera, M. Á., Alario-Hoyos, C., & Delgado Kloos, C. (2025). Evaluation of LLM tools for feedback generation in a course on concurrent programming. *International Journal of Artificial Intelligence in Education, 35*, 774–790. DOI: https://doi.org/10.1007/s40593-024-00406-0

[9] Yousef, M., Mohamed, K., Medhat, W., Mohamed, E. H., Khoriba, G., & Arafa, T. (2025). BeGrading: Large language models for enhanced feedback in programming education. *Neural Computing and Applications, 37*(2), 1027–1040. DOI: https://doi.org/10.1007/s00521-024-10449-y

[10] Zhu, E., Teja, S., Coombes, C., & Patterson, D. (2025). FEEDBOT: Formative design feedback on programming assignments. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education* (pp. 576–582). DOI: https://doi.org/10.1145/3724363.3729063

[11] Kim, D., Kim, S., & Jo, Y. (2025). Knowledge tracing in programming education integrating students' questions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 27703–27718). DOI: https://doi.org/10.18653/v1/2025.acl-long.1343

[12] Pitts, G., Hridi, A. P., & Lekshmi Narayanan, A. B. (2025). A survey of LLM-based applications in programming education: Balancing automation and human oversight. In *Proceedings of the Fourth Workshop on Bridging Human-Computer Interaction and Natural Language Processing (HCI+NLP)* (pp. 255–262). DOI: https://doi.org/10.18653/v1/2025.hcinlp-1.21

Дата першого надходження статті до видання: 22.02.2026
Дата прийняття статті до друку після рецензування: 20.03.2026
Дата публікації (оприлюднення) статті: 22.05.2026